An Intelligent Information Systems Architecture for Clinical Decision Support on the Internet

K. Canfield PhD, V. Ramesh PhD, S. Quirolgico MS, M. Silva MS
Laboratory for Healthcare Informatics, Department of Information Systems
University of Maryland, UMBC
Baltimore MD

This paper presents a prototype of an agent-based intelligent information systems architecture that can provide clinical decision support in a distributed, heterogeneous environment such as the Internet. After presenting the architecture, a specific transaction sequence is detailed and implemented to test the architecture. A transaction sequence is a detailed analysis of all actions by all entities to accomplish the system goal. In this case, the goal is to give decision support information access to a provider in the context of a computerized patient record. Based on the results of the prototype implementation, we argue that the system is scaleable and discuss other transactions, standards, and needed development.

INTRODUCTION

As Computerized Patient Records (CPRs) become more numerous at various health care centers and Internet information resources become richer, it seems clear that the two will be used together. Currently, this connection requires a large investment of time and expertise by health care providers. For example, if a provider is browsing a CPR and a question occurs to her that could be answered with Web resource, she must open a Web browser application and perform a (perhaps tedious) search for relevant information resources. In the context of today's busy managed care environment, this information seeking will probably not happen frequently. This paper describes the implementation and testing of a prototype information systems architecture for automating the scenario above. The prototype shows plausible time savings for providers who need such information. Furthermore, this architecture is shown to plausibly support much more complex transactions that are important to the future of heath care computing.

We use a design, build, and test methodology here [1]. This paper has three major objectives:

1. Specify the architecture to support distributed information systems for health care and build a prototype of it. (*Design and Build*)

2. Specify in detail a sample transaction that will take place in this architecture and test it on the prototype. (*Test*)

3. Argue from these results that the architecture is scaleable and discuss other transactions, standards, and development that are needed.

The architecture discussed here is agent-based in that there are processes running on various machines on the Internet that offer services such as resource finding, resource caching, and other coordination and control functions. The specific kind of agents used here are KQML-based (Knowledge Query and Manipulation Language) agents [2]. The remainder of this paper introduces KQML, presents the architecture and prototype, details the transaction sequence, and discusses scaleability. It is important to note that the proposed architecture is based on open standards for services and not global ownership of the system.

KQML

The Internet is a highly distributed and heterogeneous environment. The client/server model currently used on the Internet does not offer easy support of "middleware" processes for information finding, manipulation, and translation between the clients and the servers. Agents (here defined) are processes that exist on the Internet to provide these and similar services. KQML is a language that supports communications between these agents. Many KQML papers and other information are available at http://www.cs.umbc.edu/kqml.

KQML is a layered, asynchronous agent communication language (ACL) that is supported on most important transport protocols, such as tcp/ip and http. It is layered in the sense that it has a frame structure. For example, a message that contains a query to a UMLS-based vocabulary server [3] is:

0195—4210/96/$5.00

```
( ask-one
        :content (Select CUI from MRCON
                where SUI="short of breath";)
        :receiver umls-vocab-server
        :language       SQL
        :ontology       UMLS Net )
```

The "ask" line of the message is the performative. A set of performatives form the core of the language because they define the high-level interactions between agents. KQML performatives include: ask-one, ask-all, tell, subscribe, advertise, and next. The ":content" is a string from the specified ":language" that will be evaluated on the ":receiver" (server) using the ":ontology." In this case, the KQML agent gives a SQL string to a relational database server that contains UMLS tables with data conforming to the UMLS Semantic Net ontology.

Some KQML agents perform services for people and processes in organizations and other agents perform services for other agents. This later type of agent is called a facilitator. Facilitators perform services such as forwarding, brokering, recruiting, and content-based routing. For example, figure 1 shows that agent A has subscribed to agent F to monitor for the truth of x. When agent B tells F that x is true, F informs A.

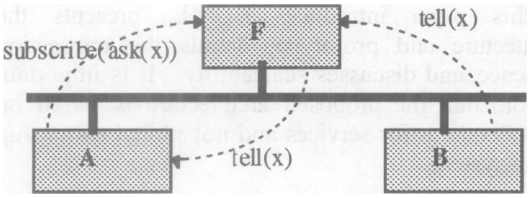

Figure 1. ACL (adapted from [2]).

If KQML were standardized for use on the Internet (or some other practical ACL), a truly distributed mediator architecture could develop without any global control of the information resources. A specific example of such an architecture is developed below.

ARCHITECTURE

The architecture developed here depends on a CPR that keeps a current active problem list, KQML speaking agents, specific knowledge sources on the Internet, and access to a Web browser. These prerequisites, with the exception of KQML speaking agents, are currently common in many health care environments. These agents, sometimes referred to as "intelligent" agents, can be built with arbitrary amounts of intelligence. The prototype described here uses rather "stupid" agents, but obtains an arguably large gain in information seeking efficiency. In a developed community of agents environment, the incentives for developing agents with more sophisticated services would increase. Our prototype architecture is shown schematically in figure 2.

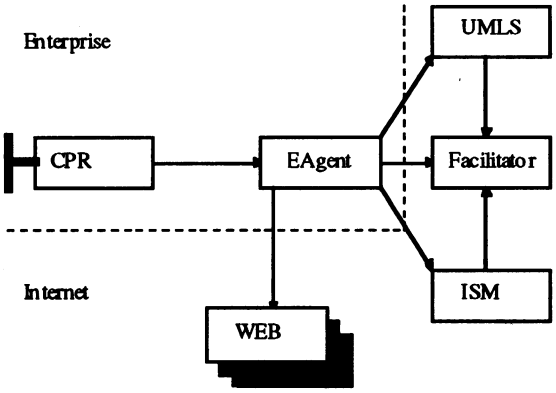

Figure 2. The Internet Agent Architecture

All entities in figure 2 have associated agents. The CPR is assumed to be on a network and able to communicate with the Enterprise Agent (Eagent) that is also running on some machine in the enterprise. This agent will keep track of things in its scope of responsibility for all patients and providers in the enterprise (say hospital or managed care organization). The Facilitator agent serves as a name server where agents can go to find available servers. The UMLS server consists of a KQML speaking agent connected to a relational database that contains (a subset of) the UMLS Metathesaurus. The ISM server consists of a KQML speaking agent connected to a relational database that contains (a subset and extension of) the UMLS Information Sources Map. The Web is merely an abstraction of all the relevant Web pages out there on the Internet. The next section details a specific transaction in this architecture.

We have implemented this architecture. The CPR (MSWindows application front-end) is currently used in a Geriatrics clinic and we have also developed a Java browser-based CPR interface. The KQML agents are written in C and are connected to Postgres object-relational databases (This is the UC Berkeley database project's object extension of Ingres which is freely available). This is a form of a federated system architecture where the KQML-speaking agents communicate with each other, but can execute any local programs. The UMLS and ISM database entities are minimally populated for this prototype. A real implementation would require fully developed Internet implementations of these entities.

TRANSACTION SEQUENCE

This transaction sequence is a fully specified use case for the proposed architecture. There are many possible such transactions. The one developed here is based on the following narrative description. The parenthetical words refer to figure 2.

A provider is browsing a patient record. The CPR provides for storing a current active problem list for each patient. Furthermore, every order that the provider makes can be linked to a current problem. Figure 3 shows such a dialog box interface where the provider can add and delete from the problem list. A personal (for the provider) agent process (CPR) on the provider's computer monitors the additions and deletions from the problem list and reports them to the enterprise agent (EAgent). In this case, the provider has added a problem to the list for a patient. The EAgent contacts the agent name server (Facilitator) to find the current names for the UMLS and ISM servers.

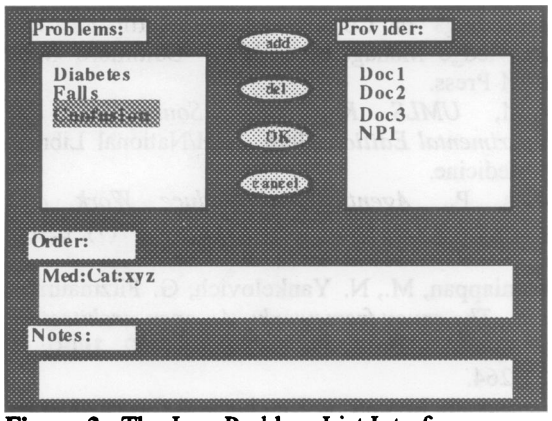

Figure 3. The Java Problem List Interface.

The EAgent then contacts the UMLS agent to ask for the canonical term for the added problem (it passes a string for a database select). Then, the EAgent asks the ISM agent for a list of URLs corresponding to the canonical form of the problem. The EAgent then edits a custom Web page for that provider/patient and adds the URLs for the new problem. At any point, the provider can request (from a menu) this information from the CPR. The Web browser will be launched and display the cached URLs as that custom Web page. The provider can then pursue the links that are sorted by problem. A similar transaction can be specified for deleting a problem.

The transaction sequence associated with this narrative is graphically shown in figure 4. The entities are shown across the top with the sequence of interactions vertically on the left side.

This transaction sequence has been implemented on the architecture. All agents and information resources perform as designed. The loop in step 5 of the sequence represents the fact that the action is performed without contacting any other agents.

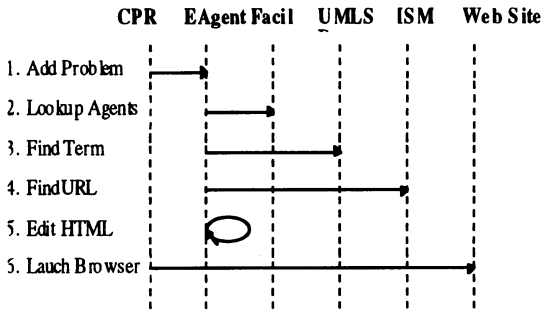

Figure 4. The Transaction Diagram.

DISCUSSION

The local agents described for this prototype project fall into the category of personal assistants [4]. Another example of an open architecture for these type of agents is Envoy [5] developed at Brown University. These are geared to the local area rather than the Internet. A similar architecture for a federated system of agents is described in Genesereth and Ketchpel from Stanford [6].

The scaleability of this kind of architecture is conceptually very good because it is open, simple, and uses existing resources with a minimum of adaptation. The devil, as always, is in the details. For this prototype project, we used minimally populated information resources because the real ones do not exist. There develops a "catch 22" here where the architecture is not developed because there are no appropriate resources and there are no resources because the architecture is not there. A plausible scenario of success is for the networking community to standardize on an ACL such as KQML to stimulate the adaptation of existing resources to the agent architecture.

There are many other beneficial transactions that could take place in this architecture. For example, assuming a transaction monitor at every CPR site, an agent could monitor every order or observation for linkage to a problem and forward the data to appropriate repositories. Since these agents keep track of where the data comes from, negotiation protocols can be developed for automatic data quality control and follow-up. Very large repositories of high quality data could be developed in this way for outcomes research and epidemiology. This ignores very difficult privacy issues that would need to be solved. The simple decision support scenario used here could also be expanded to include a system of

guideline servers that were required by different payers. In this way, providers could be easily aware of payer required treatment guidelines (which could be over-ridden and documented by provider judgment if clinically inappropriate). AHCPR currently publishes many guidelines as Web pages. Finally, updating the UMLS Metathesaurus could be semi-automated as a side effect of the canonical term lookup step in the detailed transaction. If a string could not be found, an agent could send a structured email to the responsible provider and request further information. If a term was discovered to be missing from the UMLS, a review procedure would get it added at some point.

The benefits of this architecture are increased efficiency for expensive health care workers and also increased information flow. The very simple agents developed for this project promise real time savings for providers and researchers. One future benefit of the architecture is that for small health care sites, the CPR could be offered as a remote service over the Internet. The interface could operate like the Java user interface developed for this project. Providers would contract for the interface and other value-added services supplied by agents. No expensive support operations for an on-site information system would be needed. This also assumes appropriate confidentiality and security.

The information resources are the core of the functionality and the rationale for the system of agents. The explosive growth of the Web on the Internet has stimulated the development of networked information resources, and these resources can be adapted to use the agent architecture. The agent layer will allow us to more efficiently use these resources as they grow in number and complexity. The agents described here are very simple and static. We can conceive of agents that have much more knowledge and can be configured by users and/or administrators. Finally, the structure of database processing is becoming much more distributed at the local level. As they adopt a 3-tier client/server architecture that includes a transaction monitor, they reflect the same structure as this agent architecture [7]. The middle layer becomes the perfect place for agents to live.

CONCLUSIONS

This paper has presented an architecture for a community of agents on the Internet. This intelligent information system increases access to Internet information resources while reducing the clerical and search tasks of the users. A prototype system has been implemented and successfully tested using a specific transaction sequence for clinical decision support. This prototype shows qualities of scaleability and extenseability.

References

1. Nunamaker, J., *Build and Learn, Evaluate and Learn, Informatica*, 1992, 1(1), p. 1-12.
2. Finin, T., R. Fritzson, D. McKay, R. McEntire, *KQML as an Agent Communication Language*. in proceedings of Conference on Information and Knowledge Management. 1994. Baltimore MD: ACM Press.
3. NLM, *UMLS Knowledge Sources: 7th Experimental Edition* 1996, NIH/National Library of Medicine.
4. Maes, P., *Agents that Reduce Work and Information Overload.* CACM, 1994. **37**(7): p. 31-47.
5. Palaniappan, M., N. Yankelovich, G. Fitzmaurice, *et al., The envoy framework: An open architecture for agents.* ACM Trans. Info Sys, 1992. **10**(3): p. 233-264.
6. Genesereth, M. and S. Ketchpel, *Software Agents.* CACM, 1994. **37**(7): p. 48-53.
7. Canfield, K., *Clinical resource auditing and decision support for computerized patient record systems: A mediated architecture approach.* Journal of Medical Systems, 1994. **18**(3): p. 155-166.